Laboratorio di Algoritmi e Strutture Dati

Docenti: M. Torelli, S. Aguzzoli

Appello del 9 gennaio 2008

Progetto “Same Game”
Consegna entro il 29 gennaio 2008

Il problema

Si tratta di una generalizzazione del gioco *Same Game* che nella sua versione originale si gioca su una scacchiera rettangolare di dimensioni $r \times c$. All'inizio ogni casella della scacchiera contiene una *biglia* colorata. Lo scopo del gioco consiste nel cercare di rimuovere tutte le biglie dalla scacchiera per mezzo di semplici regole che permettono di rimuovere certi insiemi di biglie dello stesso colore.

Nella generalizzazione proposta la scacchiera è il piano $\mathbb{Z}^2$ degli interi

$$\mathbb{Z}^2 = \{ (x, y) \mid x, y \in \mathbb{Z}\}$$

dove una *casella* è una coppia di interi $(x, y) \in \mathbb{Z}^2$.

Una casella (x, y) può essere vuota oppure contenere una (e una sola) biglia b di colore σ, dove σ è una parola di lunghezza arbitraria sull'alfabeto $\{a, \ldots, z\}$. Se b si trova in (x, y), diciamo che b si trova alla riga y e colonna x di $\mathbb{Z}^2$.

Le casella (x, y) di $\mathbb{Z}^2$ è *adiacente* alla casella (x', y') se e solo se (x', y') è una delle caselle

$$(x - 1, y),\ (x + 1, y),\ (x, y - 1),\ (x, y + 1)$$

Due caselle (x', y') e (x'', y'') sono *connesse* se esiste una sequenza di caselle $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ tale che:

1. per ogni $i \in \{1, \ldots, n - 1\}$, la casella (x_i, y_i) contiene una biglia,
2. $(x_1, y_1) = (x', y')$ e $(x_n, y_n) = (x'', y'')$,
3. (x_i, y_i) è adiacente a (x_{i+1}, y_{i+1}) per ogni $i \in \{1, \ldots, n - 1\}$.

Una *componente* è un insieme massimale M di caselle connesse; vale a dire che, se N è un altro insieme di caselle connesse e $M \cap N \neq \emptyset$ allora $N \subseteq M$.

Un *blocco* di colore σ è un insieme B di caselle connesse che è massimale rispetto a σ, vale a dire:

1. ogni casella di B contiene una biglia di colore σ,
2. se C è un altro insieme di caselle connesse contenenti solo biglie di colore σ e $B \cap C \neq \emptyset$, allora $C \subseteq B$.

Si osservi che un blocco B è necessariamente contenuto in un'unica componente M.

Una mossa modifica la posizione delle biglie contenute in una prefissata regione rettangolare detta *base*. Siano $x_0 \leq x_1$ e $y_0 \leq y_1$ due coppie di interi. Diciamo che una componente M è *stabile* rispetto alla *base* $([x_0, x_1], [y_0, y_1])$ se e solo se per ogni casella (x, y) di M contenente una biglia tale che $x_0 \leq x \leq x_1$ e $y_0 \leq y \leq y_1$ valgono le seguenti proprietà:

1. Se $y > y_0$, allora anche la cella $(x, y-1)$ è occupata da una biglia.
2. Se $x_0 < x \leq x_1$, la colonna $x-1$ non è *vuota* rispetto alla base, ossia: esiste almeno una casella della forma $(x-1, y)$, con $y_0 \leq y \leq y_1$, tale che $(x-1, y)$ è occupata da una biglia.

Una *mossa* consiste nello specificare una base $([x_0, x_1], [y_0, y_1])$ e una casella (x, y). L'esecuzione della mossa comporta la rimozione di tutte le biglie (x', y') tali che:

1. (x', y') appartiene al blocco cui appartiene (x, y),
2. $x_0 \leq x' \leq x_1$ e $y_0 \leq y' \leq y_1$.

Quando il blocco è tolto, occorre risistemare le biglie rimaste in modo che tutte le componenti siano stabili rispetto alla base. Sono possibili due tipi di spostamenti:

($s1$) Una biglia nella cella (x, y), con $x_0 \leq x \leq x_1$ e $y > y_0$, si sposta nella cella $(x, y-1)$, se tale cella è libera.

($s2$) Supponiamo che per qualche $x_0 < x \leq x_1$ la colonna x non sia vuota rispetto alla base e che la colonna $x-1$ sia invece vuota rispetto alla base. Allora, tutte le biglie della base nella colonna x si spostano nella colonna $x-1$ senza cambiare posizione della riga. Quindi, se una biglia si trova nella cella (x, y), con $y_0 \leq y \leq y_1$, si sposta in $(x-1, y)$.

Gli spostamenti ($s1$) e ($s2$) vanno ripetutamente applicati sulle biglie nella base fino a quando si ottieme una configurazione stabile rispetto alla base. Si noti che, indipendentemente dall'ordine in cui le biglie sono spostate, la configurazione stabile ottenuta è univocamente determinata.

Esempio

Supponiamo che le biglie siano disposte come nella figura, dove B sta per `blu`, R per `rosso` e V per `verde` (le caselle del piano non rappresentate nella figura sono vuote).

3	V	V	B	V	B	B	R	R
2	V	B	V	V	B	V	R	B
1	B	V	V	V	V	R	R	V
0	B	V	V	V	R	B	B	V
	0	1	2	3	4	5	6	7

Nel piano vi è un'unica componente connessa. Supponiamo di compiere una mossa specificata dalla base $([0, 100], [0, 20])$ e dalla casella $(2, 1)$. Occorre per prima cosa eliminare le 10 biglie nel blocco cui appartiene $(2, 1)$. Quindi occorre muovere le rimanenti nel modo specificato in modo da ottenere una componente stabile rispetto a $([0, 100], [0, 20])$. Al termine degli spostamenti si ha:

3	V				B	R	R	
2	V			B	V	R	B	
1	B	V		B	R	R	V	
0	B	B	B	R	B	B	V	
	0	1	2	3	4	5	6	7

Supponiamo ora di compiere una mossa specificata dalla base $([3,6],[1,2])$ e dalla casella $(5,1)$. Questa volta la mossa coinvolge solamente le caselle (x,y) tali che $3 \leq x \leq 6$ e $1 \leq y \leq 2$. Le biglie da rimuovere sono quelle nelle caselle $(4,1)$, $(5,1)$ e $(5,2)$. Le biglie da spostare sono quella in $(4,2)$ e quelle della colonna 6 nella base considerata. La configurazione ottenuta è:

3	V				B	R	R	
2	V			B		B		
1	B	V		B	V	V		
0	B	B	B	R	B	B	V	
	0	1	2	3	4	5	6	7

Compiendo ora la mossa specificata dalla base $([0,3],[0,2])$ e dalla casella $(0,0)$, si ottiene:

3	V				B	R	R	
2			B			B		
1			B		V	V		
0	V	V	R		B	B	V	
	0	1	2	3	4	5	6	7

e nel piano ci sono ora tre componenti connesse. Infine, compiamo la mossa specificata dalla base $([-100,1000],[-200,500])$ e dalla casella $(4,1)$. Vengono eliminate le biglie in $(4,1)$ e $(5,1)$, le altre si muovono in modo da ottenere una componente stabile rispetto a $([-100,1000],[-200,500])$. Il risultato è:

−197								
−198			B		R			
−199	V		B	B	B	R		
−200	V	V	R	B	B	V		
	−100	−99	−98	−97	−96	−95	−94	−93

Nel piano si ha una sola componente connessa e 6 blocchi.

Si richiede di implementare una struttura dati efficiente che permetta di eseguire le operazioni seguenti: (si tenga presente che la minima porzione rettangolare di piano contenente tutte le biglie può essere molto grande rispetto al numero di biglie presenti nel piano, quindi *non è sicuramente efficiente rappresentare l'insieme delle biglie mediante un'*unica *matrice*).

- **input** $(r, c, x, y, nomefile)$

  Legge dal file $nomefile$ una tabella di biglie colorate di r righe e c colonne e colloca le biglie nelle caselle corrispondenti dell'insieme $\{(x+h, y+k) \mid 0 \leq h < c,\ 0 \leq k < r\}$ secondo le regole specificate nell'apposita sezione.

- **biglia** (x, y, α)

  Pone in (x, y) una biglia di colore α, indipendentemente dal contenuto precedente di (x, y).

- **mossa** $(x, y, x_0, x_1, y_0, y_1)$

  Esegue la mossa specificata dalla base $([x_0, x_1], [y_0, y_1])$ e dalla casella (x, y).

- **componente** (x, y)

  Se (x, y) non contiene alcuna biglia allora non esegue alcuna operazione. Altrimenti, visualizza la componente contenente la casella (x, y) secondo il formato specificato nell'apposita sezione.

- **blocco** (x, y)

  Se (x, y) non contiene alcuna biglia allora non esegue alcuna operazione. Altrimenti, visualizza il blocco contenente la casella (x, y) secondo il formato specificato nell'apposita sezione.

- **numeroComponenti** ()

  Stampa il numero di componenti attualmente presenti nel piano.

All'inizio del programma il piano è vuoto. Si noti che le operazioni richieste sono liberamente implementabili; in particolare, non vanno necessariamente intese come prototipi di funzioni.

Specifiche di implementazione

Il programma deve leggere dallo standard input (`stdin`) una sequenza di righe (separate da `\n`), ciascuna delle quali corrisponde a una riga della prima colonna della Tabella 1, dove $nomefile$ è il nome di un file, α è una stringa finita sull'alfabeto $a, b, \ldots, z$ di lunghezza *arbitraria*, x, y, x_0, y_0, x_1, y_1 sono interi. I vari elementi sulla riga sono separati da uno o più spazi. Quando una riga è letta, viene eseguita l'operazione associata; le operazioni di stampa sono effettuate sullo standard output (`stdout`), e ogni operazione deve iniziare su una nuova riga.

Note

1. Non devono essere presenti vincoli sul numero di colori, blocchi, componenti, biglie e sulla lunghezza dei nomi di colori (se non quelli determinati dal tipo di dato intero). Non si richiede – anzi si sconsiglia – l'uso di grafica, se non per test personali: in modo particolare, non si usi `conio.h` e neppure `clrscr()`.

RIGA DI INPUT	OPERAZIONE
`i` $r\ c\ x\ y\ nomefile$	**input** $(r, c, x, y, nomefile)$
`b` $x\ y\ \alpha$	**biglia** (x, y, α)
`m` $x\ y\ \ x_0\ x_1\ y_0\ y_1$	**mossa** $(x, y, x_0, x_1, y_0, y_1)$
`C` $x\ y$	**componenente** (x, y)
`B` $x\ y$	**blocco** (x, y)
`n`	**numeroComponenti** $()$
`f`	Termina l'esecuzione del programma

Tabella 1: Specifiche del programma

2. Per semplicità si suppone che l'input sia sempre conforme alle specifiche di Tabella 1, per cui non è necessario controllare la correttezza dell'input. Per leggere l'input si usino le funzioni standard ANSI C `getchar()` e/o `scanf()`.

3. **Specifiche per la lettura di una tabella di biglie colorate da un file**

   Si consideri il comando:

   `i` $r\ c\ x\ y\ nomefile$.

   Allora il file di nome $nomefile$ contiene una sequenza di $r \times c$ stringhe, ciascuna delle quali è una stringa finita sull'alfabeto $\{a, b, \ldots, z\}$ oppure `0`. I colori delle biglie nel piano vengono modificati in questo modo: Per ogni $0 \leq h < c$ e ogni $0 \leq k < r$, se la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h+1)$ nel file è diversa da `0` allora si pone in $(x+h, y+k)$ una biglia di colore $\alpha_{h,k}$; se invece la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h+1)$ nel file è uguale a `0` non si modifica il contenuto della casella $(x+h, y+k)$. Si assume che la prima stringa nel file sia in posizione 1 e l'ultima sia in posizione $r \cdot c$.

   Ad esempio, se il comando è `i 2 3 4 5 tabella.txt`, e il contenuto del file `tabella.txt` è il seguente:

   ```
   blu rosso 0
   bianco 0 nero
   ```

   allora i colori delle biglie vengono modificati in questo modo:

   $(4, 5) = blu,\ (5, 5) = rosso,\ (4, 6) = bianco,\ (6, 6) = nero.$

   Nessun'altra casella viene modificata. In particolare le celle $(6, 5)$ e $(5, 6)$ non vengono toccate, in quanto le posizioni a loro corrispondenti in `tabella.txt` contengono la stringa `0`.

4. **Formato per la stampa di una componente o di un blocco**

   Si considerino i comandi:

   `C` $x\ y$ e `B` $x\ y$,

e sia $\{(x_1, y_1), (x_2, y_2), \ldots, (x_u, y_u)\}$ la componente contenente la casella (x, y). Per ogni $i = 1, \ldots, u$, sia σ_i il colore della casella (x_i, y_i). Allora l'output del comando deve essere visualizzato come segue:

(
$x_1, y_1 : \sigma_1$
$x_2, y_2 : \sigma_2$
$\vdots$
$x_u, y_u : \sigma_u$
)

L'ordine in cui vengono elencate le celle della figura è arbitrario.

Esempio

Si supponga che le righe di input siano:

```
i 6 3 -2 -4 f1.txt
i 4 5 -3 2 f2.txt
i 6 3 -4 0 f1.txt
C 1 4
B 1 4
b 1 5 rosso
B 1 4
B -2 2
n
m -2 2 -3 120 -1 4
m -2 -1 -4 120 -1 0
n
C -3 5
B -3 5
m -3 -1 -4 120 -1 1
m -3 -1 -4 120 -1 1
n
C 0 -2
B 0 -2
m -3 -1 -1000 1000 -1000 1000
m -999 -1000 -1000 1000 -1000 1000
n
C -999 -1000
f
```

dove il file `f1.txt` contiene

```
blu blu 0
0 blu giallo
0 giallo giallo
giallo rosso giallo
0 rosso 0
0 blu blu
```

mentre il file `f2.txt` contiene

```
giallo giallo giallo blu rosso
0 rosso rosso blu 0
0 0 rosso 0 rosso
giallo 0 giallo 0 blu
```

L'output prodotto dal programma deve essere il seguente

```
(
1,4:rosso
1,5:blu
)
(
1,4:rosso
)
(
1,4:rosso
1,5:rosso
)
(
-2,1:giallo
-3,2:giallo
-2,2:giallo
-1,2:giallo
-2,3:giallo
)
2
4
(
-3,5:blu
-2,5:blu
-1,5:giallo
)
(
-3,5:blu
-2,5:blu
)
6
(
-1,-2:giallo
0,-2:giallo
-1,-3:blu
0,-3:giallo
-2,-4:blu
-1,-4:blu
)
(
-1,-2:giallo
0,-2:giallo
0,-3:giallo
)
```

```
1
(
-1000,-997:giallo
-998,-997:giallo
-1000,-998:blu
-998,-998:rosso
-997,-998:blu
-1000,-999:blu
-999,-999:blu
-998,-999:rosso
-997,-999:giallo
-1000,-1000:giallo
-999,-1000:rosso
-998,-1000:giallo
-997,-1000:giallo
-996,-1000:rosso
)
```

Presentazione del progetto

Il progetto deve essere inviato per posta elettronica all'indirizzo `aguzzoli@dsi.unimi.it` entro il 29 gennaio 2008 (incluso). La discussione del progetto e l'esame orale si svolgeranno in data e luogo da specificarsi (consultare al riguardo il sito: `http://homes.dsi.unimi.it/~aguzzoli/algo.htm`).

Occorre presentare:

1. il codice sorgente (rigorosamente ANSI C, compilabile con **gcc**);
2. una sintetica relazione (formato pdf o rtf) che illustra le strutture dati utilizzate e analizza il costo delle diverse operazioni richieste dalla specifica.

I due o più file (file sorgenti C + relazione) devono essere contenuti in un unico file `.zip` il cui nome dovrà essere `cognome.zip`. La relazione e il codice devono riportare il vostro nome, cognome e matricola.

Una copia cartacea della relazione e del codice deve inoltre essere consegnata al dr. Aguzzoli entro il 29 gennaio 2008 (lasciandola eventualmente nella sua casella postale presso il dipartimento in via Comelico).

Si ricorda infine di presentarsi alla prova orale con una copia stampata della relazione e del codice.

La discussione del progetto e l'esame orale di Algoritmi e Strutture Dati si svolgeranno indicativamente nei giorni 30 gennaio, 4 e 6 febbraio 2008.

Chi abbia esigenza di sostenere la discussione orale il 30 gennaio (o comunque entro la fine di gennaio), deve segnalarlo ai docenti e consegnare il progetto entro il 28 gennaio.

Alla consegna del progetto, indicare nel testo della e-mail la data in cui si preferisce sostenere la prova orale; nei limiti del possibile si cercherà di tener conto di tali indicazioni (se non si hanno preferenze, non dare alcuna indicazione).

Il calendario degli esami orali sarà disponibile sulla pagina del corso qualche giorno dopo il termine di consegna del progetto.

Per ogni ulteriore chiarimento:
E-mail: `aguzzoli@dsi.unimi.it`
Ricevimento: il mercoledì, ore 15-16, stanza S204.

Avvisi

La versione aggiornata del progetto è pubblicata in .pdf sul sito:
`http://homes.dsi.unimi.it/~aguzzoli/algo.htm`.
Si consiglia di consultare periodicamente questo sito per eventuali correzioni e/o precisazioni relative al testo del progetto.

Si richiede allo studente di effettuare un adeguato collaudo del proprio progetto su numerosi esempi diversi per verificarne la correttezza e valutarne le prestazioni.

La realizzazione del progetto è una prova d'esame da svolgersi **individualmente**. I progetti giudicati frutto di **collaborazioni** saranno **estromessi** d'ufficio dalla valutazione.